Shruti Subramanian (ss53428)
Weihao Xu (wx924)

Computer Graphics: Mass-Spring Physical Simulation

Section 1: Goals and Key algorithms

The goal of our project is to simulate cloth. More specifically, the cloth is simulated under multiple situations, including when the cloth is hanging with two fixed corners, when winds of different directions and intensities are applied to the cloth and when a sphere interacts with the cloth. We also simulate the tearing of the cloth. The torn cloth still shows reasonable manner for all situations mentioned above.

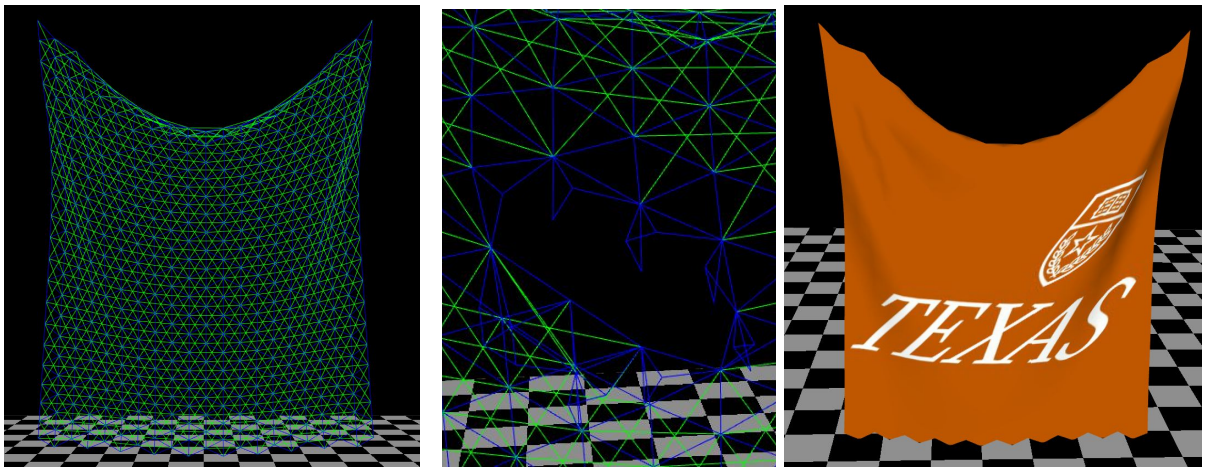
The cloth is implemented as a mass spring system shown in figure (a). It's a system consisted of particles, each having some mass, and springs, each connecting two particles. Internal and external forces might be applied to particles. Internal forces are the ones generated by the spring, which follows Hooke's law:

$$F_{internal} = -k * X$$

External forces includes gravity, wind forces, and any other forces not implemented in our project. When the system updates, all forces are summed up, and Newton's law of motion is applied to compute the velocity and position of every particle:

$$F_{total} = \sum_i F_{external_i} + F_{internal}$$

$$F_{total} = m * a$$



(a) (b) (c)

To avoid the "super elastic" artifacts seen when a spring deforms so much that the cloth no longer looks real, we implemented the constraints model [1]. In this model we constrain the deformation of each spring, making cloth deformations more reasonable.

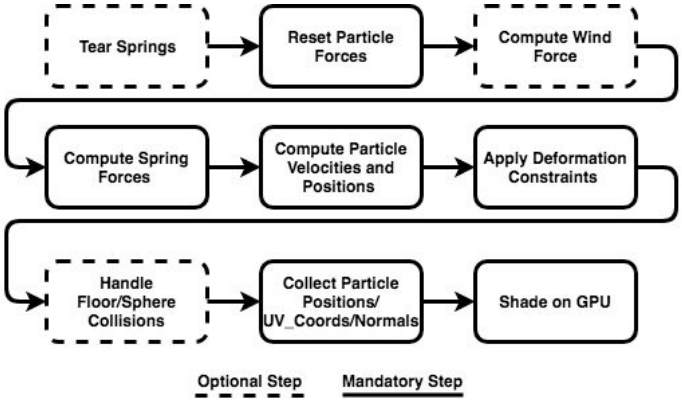
Tearing is another hard topic to address, which is shown in figure (b). We implemented a tearing algorithm similar to the approach discussed in [2]. The basic idea is to remove the selected spring and it's related springs/triangles, and to create new particles and new springs. Some corner cases are specially handled to avoid some unnatural manners.

Section 2: Implementation

Shruti Subramanian (ss53428)
Weihao Xu (wx924)

Computer Graphics: Mass-Spring Physical Simulation

We have four main data structures in our implementation of cloth - Cloth, Spring, Particle and Triangle - which allow for an implementation of the mass spring system.
A high-level workflow of one iteration of our program is:



Some implementations worth mentioning are:

- Computation of wind force: When wind is applied to a piece of cloth in different directions, the force generated vary by the directions. So when we compute wind forces, we iterate over all triangles, and compute the forces based on their face normals and wind directions:

$$F_{wind} = F_{wind0} * fabs(dot(n_{triangle}, n_{wind}))$$

- Applying deformation constraints to spring: When a spring's deformation exceeds the limitation, its length should be decreased, while maintaining the direction. This is done by moving one particle of the spring closer to the other one. We implemented a BFS algorithm for the "layer by layer" propagation: we set the current particle's position, and adjust all its' neighbors' positions so that the spring length meets our requirements. Since the two corner particles are fixed to hold our cloth, they will always be pushed to the queue for BFS first.

A known bug in our system is seen with collision detection. Our collision detection algorithm looks at collisions between particles and an object (either the sphere or floor). Since we didn't use a physically correct model, artifacts can be seen in some situations (In our case, it is especially seen when wind is turned on). One known way to fix this would be to add more vertices to our cloth such that the triangles have smaller faces and the artifacts become irrelevant. However, given our compute power/time to run our model we decided not to pursue this route.

Reference:

[1] Provot, Xavier. "Deformation constraints in a mass-spring model to describe rigid cloth behaviour." *Graphics interface*. Canadian Information Processing Society, 1995.
[2] Onal, Emre, and Veysi Isler. *Cloth Tearing Simulation*. Diss. Master's thesis, Graduate School of Informatics of the Middle East Technical University, 2013.